Image To Captions

Release 00.00.01

Alessandro Goller & Riccardo Ricci

May 02, 2023

CONTENTS:

CHAPTER

ONE

APP

1.1 app package

1.1.1 Subpackages

app.config package

Submodules

app.config.config module

Module with all config

class app.config.config.**Config**(_env_file: str | PathLike | List[str | PathLike] | Tuple[str | PathLike, ...] | None = '<object object>', _env_file_encoding: str | None = None, _env_nested_delimiter: str | None = None, _secrets_dir: str | PathLike | None = None, *, SQLALCHEMY_DATABASE_URI: str = 'postgresql://AlessandroGoller:Xwx7lcqU8pgj@ep-lively-snow-111074.eu-central-1.aws.neon.tech/neondb', LANGUAGE: str = 'italian', PORT: str = '5000', HUGGINGFACEHUB_API_TOKEN: str | None = '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', OPENAI_API_TOKEN: str | None = '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', SERPAPI_API_KEY: str | None = 'd3a3fb9f0060ab6819eb3dc743ab7e9f8f70ccef046942779650d18a72bafdd0', REPLICATE_API_KEY: str | None = None, MODEL_BLIP: str | None = 'andreasjansson/blip-2:4b32258c42e9efd4288bb9910bc532a69727f9acd26aa08e175713a0a857a608')

Bases: `BaseSettings`

Class with the config

HUGGINGFACEHUB_API_TOKEN: str | None

LANGUAGE: str

MODEL_BLIP: str | None

OPENAI_API_TOKEN: str | None

PORT: str

REPLICATE_API_KEY: str | None

SERPAPI_API_KEY: str | None

SQLALCHEMY_DATABASE_URI: str

Module contents

app.config Init

app.crud package

Submodules

app.crud.company module

Module for crud for Companies

`app.crud.company.`**`create_company`**(*company: CompanyCreate*) → Company | None

Creation a user, in input the schema of user create and return the user

`app.crud.company.`**`delete_company`**(*company: Company*) → dict[str, bool]

" Permit to delete a company

`app.crud.company.`**`get_company_by_id`**(*id_company: int*) → Company | None

Return the Company from id_company

`app.crud.company.`**`get_company_by_name`**(*name: str*) → Company | None

return Company from email

`app.crud.company.`**`get_company_by_user_id`**(*user_id: int*) → Company | None

Return the Company from user_id

`app.crud.company.`**`update_company`**(*company: Company*, *company_edit: CompanyInfoBase*) → Company | None

Permit to edit info inside company model

app.crud.instagram module

Module for crud for Instagram

`app.crud.instagram.`**`bulk_create_instagram`**(*instagrams: list[app.schema.instagram.InstagramCreate]*) → None

Bulk creation of instagrams

`app.crud.instagram.`**`create_instagram`**(*instagram: InstagramCreate*) → Instagram | None

Creation a instagram, in input the schema of instagram create and return the instagram

`app.crud.instagram.`**`delete_instagram`**(*instagram: Instagram*) → dict[str, bool]

Permit to delete a instagram

`app.crud.instagram.`**`get_instagram_after_date`**(*company_id: int*, *date: datetime*) → list[app.model.instagram.Instagram] | None

Return the list of Instagram posts from company_id with date after the input date `app.crud.instagram.`**`get_instagram_by_company_id`**(*company_id: int*) → list[app.model.instagram.Instagram] | None

Return the list of Instagram from company_id

`app.crud.instagram.`**`get_instagram_by_id`**(*id_instagram: int*) → Instagram | None

Return the Instagram from id_instagram

`app.crud.instagram.`**`get_instagram_by_url`**(*url: int*) → Instagram | None

Return the Instagram from url

`app.crud.instagram.`**`get_instagram_by_user_id`**(*user_id: int*) → list[app.model.instagram.Instagram] | None

Return the list of Instagram from user_id

`app.crud.instagram.`**`get_last_n_instagram`**(*company_id: int*, *number_ig: int*) → list[app.model.instagram.Instagram] | None

Return the list of n Instagram from company_id order by date

`app.crud.instagram.`**`insert_data_to_db`**(*data: dict*, *user_id: int*, *company_id: int*) → bool

From a dict of data, insert everythin inside Instagram db

`app.crud.instagram.`**`update_instagram`**(*instagram: Instagram*, *instagram_edit: InstagramInfoBase*) → Instagram | None

Permit to edit info inside instagram model

app.crud.user module

Module for crud for Users

`app.crud.user.`**`become_admin`**(*email: str*) → User | None

Convert a user to admin

`app.crud.user.`**`create_user`**(*user: UserCreate*) → User | None

Creation a user, in input the schema of user create and return the user

`app.crud.user.`**`get_user_by_email`**(*email: str*) → User | None

return user from email

`app.crud.user.`**`get_user_by_id`**(*user_id: str*) → User | None

Return the user from user_id

`app.crud.user.`**`get_users`**() → list[app.model.user.User] | None

Return the list of users

Module contents

Module Init

app.model package

Submodules

app.model.company module

Module Model Company

`class app.model.company.Company(**kwargs)`

Bases: `Base`

Class for Company Model

`description`

`id_company`

`id_user`

`name`

`url_instagram`

`website`

app.model.instagram module

Module Model Instagram

`class app.model.instagram.Instagram(**kwargs)`

Bases: `Base`

Class for Instagram Model

`comments`

`date`

`hashtags`

`id_company`

`id_instagram`

`id_user`

```
idx_instagram_unique_cols = Index('idx_instagram_unique_cols', Column('id_company',
Integer(), table=<t_instagram>, nullable=False), Column('id_user', Integer(),
table=<t_instagram>, nullable=False), Column('post', String(), table=<t_instagram>),
Column('image_description', String(), table=<t_instagram>), Column('hashtags',
String(), table=<t_instagram>), Column('mentions', String(), table=<t_instagram>),
Column('tagged_users', String(), table=<t_instagram>), Column('likes', Integer(),
table=<t_instagram>), Column('comments', Integer(), table=<t_instagram>),
Column('date', DateTime(), table=<t_instagram>), Column('location', String(),
table=<t_instagram>), Column('typename', String(), table=<t_instagram>),
Column('mediacount', Integer(), table=<t_instagram>), Column('title', String(),
table=<t_instagram>), Column('posturl', String(), table=<t_instagram>), unique=True)
```

image_description

likes

location

mediacount

mentions

post

posturl

tagged_users

title

typename

app.model.user module

Module Model User

class app.model.user.**User**(***kwargs*)

Bases: Base

Class for User Model

admin

email

last_access

name

password

time_created

user_id

Module contents

Module Init

app.schema package

Submodules

app.schema.company module

Module for Company Schema

class app.schema.company.**CompanyCreate**(*, *name: str*, *url_instagram: str | None = None*, *description: str | None = None*, *website: str | None = None*, *id_user: int*)

Bases: `CompanyInfoBase`

Class CompanyCreate Use it during creation

id_user: int

class app.schema.company.**CompanyInfoBase**(*, *name: str*, *url_instagram: str | None = None*, *description: str | None = None*, *website: str | None = None*)

Bases: `BaseModel`

Class CompanyInfo

description: str | None

name: str

url_instagram: str | None

website: str | None

app.schema.instagram module

Module for Instagram Schema

class app.schema.instagram.**InstagramCreate**(*, *post: str | None = None*, *image_description: str | None = None*, *hashtags: str | None = None*, *mentions: str | None = None*, *tagged_users: str | None = None*, *likes: int | None = None*, *comments: int | None = None*, *date: datetime | None = None*, *location: str | None = None*, *typename: str | None = None*, *mediacount: int | None = None*, *title: str | None = None*, *posturl: str | None = None*, *id_company: int*, *id_user: int*)

Bases: `InstagramInfoBase`

Class InstagramCreate Use it during creation

id_company: int

id_user: int

class app.schema.instagram.**InstagramInfoBase**(*, *post: str | None = None*, *image_description: str | None = None*, *hashtags: str | None = None*, *mentions: str | None = None*, *tagged_users: str | None = None*, *likes: int | None = None*, *comments: int | None = None*, *date: datetime | None = None*, *location: str | None = None*, *typename: str | None = None*, *mediacount: int | None = None*, *title: str | None = None*, *posturl: str | None = None*)

Bases: `BaseModel`

Class InstagramInfo

comments: int | None

date: datetime | None

hashtags: str | None

image_description: str | None

likes: int | None

location: str | None

mediacount: int | None

mentions: str | None

post: str | None

posturl: str | None

tagged_users: str | None

title: str | None

typename: str | None

app.schema.user module

Module for User Schema

class app.schema.user.**UserCreate**(*, *name: str*, *email: str*, *password: str*)

Bases: `UserInfoBase`

Class UserCreate Use it during creation

password: str

class app.schema.user.**UserInfo**(*, *name: str*, *email: str*, *user_id: int*)

Bases: `UserInfoBase`

Class UserInfo Use it during retrieving information

class Config

Bases: `object`

orm_mode = True

user_id: int

class app.schema.user.**UserInfoAdmin**(*, *name: str*, *email: str*, *user_id: int*, *admin: bool*)

Bases: `UserInfoBase`

Class UserInfoAdmin

class Config

Bases: `object`

orm_mode = True

admin: bool

user_id: int

class app.schema.user.**UserInfoBase**(*, *name: str*, *email: str*)

Bases: `BaseModel`

Class UserInfoBase

email: str

name: str

Module contents

Module Init

app.services package

Submodules

app.services.ig_scraping module

class app.services.ig_scraping.**GetInstagramProfile**

Bases: `object`

Class to load a profile and download its data.

download_hastag_posts(*hashtag*)

Download posts that contains a certain hashtag.

download_users_posts_with_periods(*username*)

Download posts from a user within a specified period of time.

download_users_profile_picture(*username*)

Download a user's profile picture.

get_post_comments(*username*)

Function to get a post's comments and print them at screen.

get_post_info_csv(*username*)

Function to get info from every post of a user and store them in a csv file.

get_post_info_json(*username*, *last_n_posts=100*)

Function to get info from every post of a user and store them in a json file (dictionary style).

get_users_followers(*user_name*)

Function to get a profile's followers. Note: login required to get a profile's followers.

get_users_followings(*user_name*)

Function to get a profile's followings. Note: login required to get a profile's followings.

app.services.ig_scraping.**load_post_captions_from_json**(*json_file: str*, *shortcodes: list | None = None*) → list

Function to load post captions from a json file. The post loaded are the one specified by the shortcodes list. This to allow compatibility with eventual search functions on the specific post.

app.services.langchain module

Module for langchain

`app.services.langchain.`**`generate_ig_post`**`(`*prompt: str*`)` → str

> Function to generate a post for Instagram using a predefined prompt and chatgpt

`app.services.langchain.`**`generate_img_description`**`(`*image: BytesIO*, *model: str = 'andreasjansson/blip-2:4b32258c42e9efd4288bb9910bc532a69727f9acd26aa08e175713a0a8*
→ str

> Function to generate a description of an image using blip2

`app.services.langchain.`**`prepare_llm`**`()` → HuggingFaceHub

> Return the llm

`app.services.langchain.`**`search_info_of_company`**`(`*name_to_search: str*`)` → str

> Search on Internet for info of a company name and return them.
>
> **Parameters**
> **`name_to_search`** (*str*) –
>
> **Return type**
> str

Module contents

App Init

app.utils package

Subpackages

app.utils.streamlit_utils package

Submodules

app.utils.streamlit_utils.auth module

Module utils for streamlit auth

`app.utils.streamlit_utils.auth.`**`is_logged_in`**`(`*session: dict[str, bool]*`)` → bool

> Verifica se l’utente è già loggato o meno.

`app.utils.streamlit_utils.auth.`**`register_user`**`(`*email: str*, *password: str*`)` → bool

> Registra un nuovo utente con email e password. Restituisce True se la registrazione è andata a buon fine, False se l’username è già stato utilizzato.

`app.utils.streamlit_utils.auth.`**`verify_login`**`(`*email: str*, *password: str*`)` → bool

> Verifica se l’utente con username e password esiste nella lista degli utenti registrati e se le credenziali sono corrette.

Module contents

app.utils Init

Submodules

app.utils.decorators module

Decorators Module

`app.utils.decorators.`**`timed_lru_cache`**(*timeout: int*, *maxsize: int = 128*, *typed: bool = False*) → Any

Extension of functools lru_cache with a timeout

Parameters: timeout (int): Timeout in seconds to clear the WHOLE cache, default = 10 minutes maxsize (int): Maximum Size of the Cache typed (bool): Same value of different type will be a different entry

Example: @timed_lru_cache(maxsize=12, timeout=15)

`app.utils.decorators.`**`timeit`**(*f: Callable*) → Callable

Decorator to measure time

app.utils.logger module

Module to configure logger

`app.utils.logger.`**`configure_logger`**`()` → Any

Settings Logger

Module contents

app.utils Init

1.1.2 Submodules

1.1.3 app.dependency module

Module with the dependency

`app.dependency.`**`get_db`**`()` → Iterator

Return an iterator to the db

`app.dependency.`**`get_settings`**`()`

Return the config

1.1.4 app.main module

Main file

`app.main.main()` → None

Prepare DB

1.1.5 app.streamlit_app module

Module for Streamlit

`app.streamlit_app.main()` → None

Main streamlit

1.1.6 Module contents

App Init

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX